

Протокол до Домашньої роботи №2



HAZELCAST

Андрій Полішко

Distributed map

1. Створення та запис до розподіленої мапи

```
1 import hazelcast
2
3 client = hazelcast.HazelcastClient(
4     cluster_name="hello-world",
5 )
6
7 # Create a Distributed Map in the cluster
8 map = client.get_map("my-distributed-map").blocking()
9
10 values = ['BMW', 'Mercedes', 'Audi', 'Toyota', 'Honda', 'Ford', 'Nissan', 'Tesla', 'Hyundai', 'Kia']
11
12 for i in range(1000):
13     map.put(str(i), values[i % len(values)])
```

2. Розподіл даних у Management Center

Map Statistics (In-Memory Format: BINARY) RESET TIME 1 minute ago → now Default View 🗑️ 📄 +

| Member ^ | Entries | Gets | Puts | Removals | Sets | Entry Memory | Events |
|----------------|---------|------|-------|----------|------|--------------|--------|
| 127.0.0.1:5701 | 338 | 0 | 345 | 0 | 0 | 43.55 kB | 0 |
| 127.0.0.1:5702 | 319 | 0 | 319 | 0 | 0 | 41.14 kB | 0 |
| 127.0.0.1:5703 | 343 | 0 | 345 | 0 | 0 | 44.21 kB | 0 |
| TOTAL | 1,000 | 0 | 1,009 | 0 | 0 | 128.90 kB | 0 |

1 - 3 of 3 Rows 10 ▾

3. Зміни розподілу за різних обставин

а. При відключенні однієї ноди(третьої)

Map Statistics (In-Memory Format: BINARY) RESET TIME 1 minute ago → now Default View 🗑️ 📄 +

| Member ^ | Entries | Gets | Puts | Removals | Sets | Entry Memory | Events |
|----------------|---------|------|------|----------|------|--------------|--------|
| 127.0.0.1:5701 | 514 | 0 | 345 | 0 | 0 | 66.21 kB | 0 |
| 127.0.0.1:5702 | 486 | 0 | 319 | 0 | 0 | 62.69 kB | 0 |
| TOTAL | 1,000 | 0 | 664 | 0 | 0 | 128.90 kB | 0 |

1 - 2 of 2 Rows 10 ▾

б. При відключенні двох (другої та третьої)

Map Statistics (In-Memory Format: BINARY) RESET TIME 1 minute ago → now Default View 🗑️ 📄 +

| Member ^ | Entries | Gets | Puts | Removals | Sets | Entry Memory | Events |
|----------------|---------|------|------|----------|------|--------------|--------|
| 127.0.0.1:5701 | 1,000 | 0 | 345 | 0 | 0 | 128.90 kB | 0 |
| TOTAL | 1,000 | 0 | 345 | 0 | 0 | 128.90 kB | 0 |

1 - 1 of 1 Rows 10 ▾

Як бачимо на останньому скріншоті, втрати даних не відбулось. Це

досягається за допомогою реплікації.

Distributed map with and without locks

1. Значення при інкременті 10000 разів трьома клієнтами без механізмів синхронізації одночасно

Map Browser

Key

1000

Key Type

String

Need to enable per entry stats of the map to see all the values. Please see the [documentation](#)

| | |
|-------------------|-------------------|
| Value: | 25820 |
| Memory Cost: | 64.00 B |
| Expiration Time: | N/A |
| Last Access Time: | N/A |
| Last Stored Time: | N/A |
| Time to Live: | Unlimited |
| Key Owner Member: | 127.0.0.1:5703 |
| Class: | java.lang.Integer |
| Creation Time: | N/A |
| Hits: | N/A |
| Last Update Time: | N/A |
| Version: | 120014 |
| Max Idle: | Unlimited |

Cancel

BROWSE

Як бачимо, значення не дорівнює 30000, через race condition.

Код використаний для цього прикладу знаходиться у теці
`./raceConditionTest/notLockedExamples`

2. Аналогічний тест з використанням блокування нашої змінної (Pessimistic locking)

Map Browser

Key

1000

Key Type

String

Need to enable per entry stats of the map to see all the values. Please see the [documentation](#)

| | |
|-------------------|-------------------|
| Value: | 30000 |
| Memory Cost: | 64.00 B |
| Expiration Time: | N/A |
| Last Access Time: | N/A |
| Last Stored Time: | N/A |
| Time to Live: | Unlimited |
| Key Owner Member: | 127.0.0.1:5703 |
| Class: | java.lang.Integer |
| Creation Time: | N/A |
| Hits: | N/A |
| Last Update Time: | N/A |
| Version: | 160016 |
| Max Idle: | Unlimited |

Cancel
BROWSE

Як і очікувалося, значення дорівнює 30000, тому що при кожній зміні значення, воно блокувалось поточним клієнтом (це також відобразилось на швидкодії, такий код був відчутно повільніший)

Код використаний для цього прикладу знаходиться у теці

./raceConditionTest/pessimisticLockedExamples

3. При оптимістичному блокуванні результат, очікувано, аналогічний

Код використаний для цього прикладу знаходиться у теці

./raceConditionTest/optimisticLockedExamples

Bounded queue

[illegible]

Результати декількох запусків одного продюсера та двох конс'юмерів

Так як ми поставили побмеження на 10 елсентів до нашої черги, то відповідно запускати продюсера потрібно декілька разів, тому що кожен консюмер очікує по 100 елементів.

Конс'сюмери забирають елементи з черги по черзі, якщо запуснені паралельно.

Якщо консюмери не запуснені, а черга переповнилась, то просто не додаються нові елементи.

Посилання

https://github.com/AndriiPolishko/APZ_labs/tree/main/Lab2